

CodeVoting

Protection Against Automatic Vote Manipulation in an Uncontrolled Environment

Rui Joaquim and Carlos Ribeiro

ISEL / INESC-ID
rjoaquim@cc.isel.ipl.pt,
carlos.ribeiro@tagus.ist.utl.pt

Abstract. One of the major problems that prevent the widespread of Internet voting is the vulnerability of the voter's computer. A computer connected to the Internet is exposed to virus, worms, spyware, malware and other threats that can endanger the election's integrity. For instance, it is possible to write a virus that changes the voter's vote to one predetermined vote on election's day. It is possible to write such a virus so that the voter would not notice anything wrong with the voting application. This attack is very dangerous because it may pass undetected. To prevent such attack it is necessary to prevent automatic vote manipulation at voter's computer. Here we present CodeVoting, a technique to create a secure communication channel to a smart card that prevents vote manipulation by the voter's PC, while at the same time allows the use of any cryptographic voting protocol to protect the election's integrity at the server side of the voting application.

Keywords: Internet voting, vote manipulation.

1 Introduction

Remote electronic voting can be a powerful tool for our democracies. It can allow citizens to vote from anywhere at anytime and also provides faster vote count. However, it also brings some risks. Risks to the integrity of the election such as the risk of automatic vote manipulation, and risks to the privacy of voters that could ruin one of the pillars of our democracies. These risks apply both to the client and server side of the voting application.

Currently, cryptographic mechanisms can be employed to provide protection at server side. Techniques, such as digital signatures and zero knowledge proofs are normally used to guarantee correct vote handling. To protect voters' privacy other cryptographic techniques are used such as homomorphic ciphers, mix-nets and blind signatures.

Cryptographic voting protocols work at server side based on the assumption that several parties do not collude. On the other hand we have the client side of the voting application. The client application is centralized, i.e. not divided over several opposite entities, therefore it is usually considered "trusted". The

client application shows the ballot, collects the answer, performs the voting protocol and verifies it on behalf of the voter. However, there is a problem with this approach, and the problem is considering the client application “trusted”. The client application can be manipulated to cast a vote on a predetermined candidate while it misleads the voter into believing that her vote is the one that is cast. Another weak point at client’s side is the possibility of compromising the client application, or the remote computer where the client program executes, to facilitate vote buying and coercion. If the voter uses a computer that is controlled by the vote buyer or coercer it is easy to produce a vote receipt.

There is much work on the vote buying/coercion problem [4,5,9,10,11,6,12,15] under the assumption of a secure voter’s platform. However, there is few on the insecurity of voter’s platform. The work on vote buying and coercion free voting systems follows two main approaches. The first one is the use of a secure and secret communication channel between the voter and a trusted party of the voting system, allowing the voter to cheat on the buyer/coercer. Depending on the voting system, such secure and secret communication channel can be required prior to or on the election day. The second main technique used to prevent vote buying/coercion is allowing the voter to vote several times. Therefore, since usually the list of voters who voted is public, the only real vote buying/coercion possible that gives the attacker 100% of guarantees is to buy/coerce the voter to abstain.

We understand that vote buying/coercion is a potential big problem on Internet voting systems. However, we consider that the use of an insecure voters’ platform can have a potential higher risk to election’s integrity. We base our opinion on three reasons.

- First, large scale vote buying/coercion, involving possibly thousands of voters, is quite unlikely to pass undetected.
- Second, with all the security flaws on operating systems and applications, it is easy to write a virus that would be active on election’s day to change the voter’s vote.
- Third, we believe that writing a virus and disseminating it would be cheaper and more difficult to trace back to the authors than a vote buying/coercion attempt of a thousand voters, therefore more appealing to an attacker.

Therefore, we can say that the client side weaknesses of a remote electronic voting system is the main issue that prevents the widespread of remote electronic voting [2,7,8,13].

CodeVoting addresses some problems of the insecure voter’s platform, namely automatic vote manipulation. CodeVoting works by creating a secure communication channel to a trusted component of the voting system. Therefore, CodeVoting can be considered as a kind of user interface to the voting system.

We propose the use of a tamper resistant device, such as a smart card, to be the trusted component of the voting system at client’s side. Since a smart card provides a much more secure execution platform than an off-the-shelf PC, using CodeVoting to create a secure communication channel to the smart card,

throughout the voter's PC, will prevent automatic vote manipulation by malicious software installed in the voter's PC. Nevertheless, the voter's PC network and I/O capabilities will still be used to interact with the voter and the server side of the voting system.

The use of secure devices, e.g. smart cards, in electronic voting is not new [5,11]. Secure devices are typically used as a way to provide secure voter's authentication and for the generation and secure storage of secret values of the voting protocol. However, there is always the need to use the voter's computer to show the ballot and collect the answer, and usually this is assumed to be performed by a trusted vote client application. Our goal is to show how one can build a simple, secure and private communication channel between the common voter and her smart card, without the need to trust on the voter's computer. Secure communications channels are easy to achieve between machines, e.g. by sharing a secret key. However, making a secure and private communication channel between a machine and a common human being is not so straightforward. The challenge is to keep the complexity of the communication channel as small as possible so the human can deal with it.

1.1 Vote Manipulation Attacks

A vote manipulation attack can be a modification of the voter's vote. The vote modification attack can be performed in two ways: i) changing the vote to a predetermined candidate or ii) changing the vote to a random candidate. While the first attack is more powerful the second may be easier to prepare in advance. By other words, to change a vote to a predetermined candidate one must have the knowledge of which candidate one wants to change the vote to, while to change the vote to a random candidate there is no need to know the candidates in advance.

If one wants to boost the number of voter for candidate A it is preferable to perform an attack to directly change the votes to votes for candidate A. On the other hand, if one wants to decrease the votes for candidate B, it suffices to perform a random vote modification attack in an area known to be much favourable to candidate B.

The other kind of vote manipulation attack is to fake a successful vote delivery. In many voting systems this can be done just by presenting the message "Your vote was successfully delivered. Thank you for voting.". This attack allows an attacker to reduce the votes on a candidate just by targeting an area with great affinity for that candidate.

In the next section we describe current proposals to minimize the weaknesses at the client side. In Sect. 3 we present Code Voting, a solution that prevents vote manipulation at client side. Then, we present in more detail the main components of CodeVoting in Sect. 4 and 5. We evaluate CodeVoting resistance to vote manipulation attacks on Sect. 6. Finally, we present some issues and future work on CodeVoting in Sect. 7 and conclude in Sect. 8.

2 Related Work

Cryptographic voting protocols can prevent vote manipulation at server side but that's only relevant if those properties cannot be easily broken at the uncontrolled client side of the remote voting system. Here we present an overview of the current proposals to deal with this problem.

One proposal is to restrict remote electronic voting to controlled environments [2], such as controlled voting kiosks, providing immediate protection on the common threats of uncontrolled environments, such as virus and other malicious programs. In a controlled environment it is also possible for election officials to verify that the correct client application is installed and running. However, this solution as the disadvantage of restricting voter's mobility and does not really work in the presence of corrupted voting officials who can install a malicious vote client application.

In 2001, Chaum [3] presented SureVote. SureVote allows the voter to vote using secret vote and reply codes. These secret codes allow the voter to detect if anyone changed the vote code during the voting process. SureVote consists in the generation of a secret vote and reply code for each candidate and for each voter. The codes are delivered to the voters prior to the election day. On election day the voter sends the vote code of her favourite candidate through the voting channel, e.g. Internet. At server side the reply code is computed by a set of trustees and sent to the voter that confirms it to verify that there was no vote modification. After the election end the trustees compute the real votes from the vote codes and publish the results. However, if there is at least one corrupted trustee, SureVote does not guarantee that in the counting phase the vote code is translated to the right candidate. Another issue with SureVote is that to protect against vote tracking and personification it is necessary to secretly create and anonymously deliver the vote codes to the voters, a requirement that is very hard to achieve and that creates new opportunities to attack the voting system.

A SureVote similar voting scheme was also used in the UK to enable the use of Internet, SMS and telephone voting channels [16]. The drawbacks of this system are the same than the ones of SureVote, i.e. it is necessary to guarantee that the codes are secretly generated and anonymously delivered to the voters, and there is no guarantees that the code vote is translated to the right candidate. The reply code only confirms that the vote has reached an entity that knows the right reply code.

Another proposal to solve the insecure platform problem is to use trusted computing technology [17]. Trusted computing is a technology that allows remote attestation of machines and programs running on them. With remote attestation it is possible to certify that the voter is using the correct voting program. Trusted computing also provides ways to secure I/O operations between the program and the physical I/O devices, therefore creating a secure environment for an application to run. The attestation process is based on measures performed on the software by a hardware module called trusted platform module (TPM). The client of a remote voting application needs to interact with the voter (I/O device drivers), needs to establish a connection with the voting server (network

protocol stack + network adapter driver) and, last but not least, it needs an environment to run on, i.e. what it really needs is a working operating system. The attestation of the core of the operating system, the device drives and the voting application can be cumbersome. Moreover, there are also problems concerning the maturity of the currently deployed technology [14] and concerning the revocation of cracked machines [1]. We believe that, for now, the application of trusted computing to remote voting as the only guaranty of correct application behaviour is not a valid alternative.

Kutyłowski and Zagórski [10] propose the use of cryptographically hidden ballots. The voter requests n encrypted ballots that are randomized. Then she chooses one to use and verifies the other ballots with the help of a private external channel. If the revealed ballots are correct the voter uses the chosen ballot to cast her vote. The main disadvantages of this solution are the need of a private external channel to verify the ballots and the complexity of the voting protocol that requires the voter to deal directly with cipher texts.

The last proposal we are aware of replaces the standard PC's I/O by a secure I/O device [18]. The disadvantages of this solution are the necessity of a non standard I/O device, the corresponding costs, and the reduced I/O capabilities of the device used. However, it may be a possibility in a semi-controlled environment as a solution that allows adapting a standard PC to Internet voting.

3 CodeVoting

Reading the last section the reader quickly understands that none of the presented proposals is perfect. Here we present a proposal that does not restrict the voter's mobility, protects the vote even if the vote client application is executed in a malicious machine, and allows for the use of cryptographic voting protocols that protect voter's privacy and vote integrity at server side. Nevertheless, our system is simple enough to be used by the common voter.

Briefly, our solution consists in the following steps: i) the voter expresses her vote as a secret code, ii) the secret code is translated into the corresponding candidate code (clear vote), iii) the clear vote is used in a cryptographic voting protocol.

We propose the use of a tamper resistant device, such as a smart card, to be the trusted component of the voting system at client side. This trusted component we will call from now on VoterCard. The VoterCard will be used to securely authenticate the voter to the vote server, e.g. by means of digital signature, and it will also be in charge of i) the translation of the secret code to the candidate code, hiding the clear vote from the vote client application, and ii) the use of the cryptographic mechanisms provided by the cryptographic voting protocol. Additionally, it also provides proof of correct conclusion of the voting protocol.

3.1 CodeVoting Details

CodeVoting can be seen as a rearrangement of the ideas presented by Chaum [3]. However, the idea of CodeVoting is to use the codes just as a user interface and

not as the entire voting protocol. The secret codes are the base for the secure and private communication channel between the voter and her VoterCard. The voter will use secret codes to choose her favorite candidate. Each VoterCard has a set of secret codes associated with it that are printed on a paper card, the CodeCard. The details on how the voter gets her VoterCard and CodeCard are explained later.

For the voter the voting process is quite simple. The voter just uses a CodeCard to translate the candidate code into a vote code.

Election for the Most Important Figure in Security A - Alice B - Bob C - Eavesdropper D - Attacker Enter your vote code:	CodeCard	
	Candidate	Vote Code
	Blank vote	SIT5Y
	A	A3CR2
	B	97RG7
	C	GHFT1
	D	WL764
	...	
	Confirmed vote delivery code 6HKG2	

Fig. 1. Example of a ballot (on the left) and a CodeCard (on the right)

For example, a voter, with the ballot and CodeCard of Fig. 1, that wishes to vote for candidate D just have to enter WL764 as the vote code.

Every voter will have a different CodeCard, therefore different vote codes for the same candidate. Each CodeCard is associated to a VoterCard, which is responsible for the translation of the vote code to the candidate code. Only the voter and the VoterCard should know the codes written on the CodeCard. Therefore, CodeVoting protects against a malicious voting application trying to change the voter’s vote.

After translating the vote code to the candidate code any voting protocol can be used to cast the vote. One solution is using the VoterCard to process the candidate code accordingly to the cryptographic voting protocol and submit the vote. Another alternative is to create an unchangeable vote and pass it to the voting client application running on the voter’s PC so it can proceed with the voting protocol.

When the VoterCard receives a confirmation of a successful vote delivery it releases the confirmed vote delivery code, assuring the voter that her vote was successfully delivered.

Based on this overview of CodeVoting the reader can easily understand that CodeVoting is a kind of an user interface plugin to a voting system that can protect the voter’s choice from manipulation.

4 VoterCard

As explained before, the VoterCard is in charge of the translation of the vote code to the clear vote, and also of the execution of a cryptographic voting protocol. However, we did not explained how each voter gets a VoterCard.

We propose to do the distribution of the VoterCards to the voters in a registration phase. This procedure is only required once, i.e. the VoterCard will be reused in subsequent elections. Since the VoterCard is a smart card we propose the use of it also as a secure voter's authentication mechanism, by means of digital signature. Therefore, a public key infrastructure (PKI) should be in place before the registration process. The PKI used can be set up just for elections' proposes or can be of more wide-use in a national e-Government project. This last approach can be useful to prevent at some level vote buying and coercion, because if the voter gives her VoteCard to a vote buyer/coercer it is not just a vote that the voter gives away, it is also all the e-Government rights of the voter.

4.1 Is the VoterCard Trustworthy?

Reading the description of CodeVoting, the reader quickly understands that CodeVoting relies on the correct behaviour of the VoterCard. Therefore one can ask: CodeVoting is designed to protect the voter from the insecure voter's PC, but what guarantees are given that the VoterCard does in fact do what it is supposed to do? One way to verify that the VoterCard does in fact what it is supposed to do is to test it. Besides testing the VoterCards in the production phase we believe that would be good to have additional random testing by an independent certification authority.

Additionally we can make voters also part of the certification process. It could be possible for a voter to verify her VoterCard by running a fake election with instant results sometime before the real election.

Nevertheless, one can point out that the application running inside the VoterCard is somehow able to detect that is being subject to a test, and therefore it will act properly in the tests but it will still change the voter's vote on the real election day. To prevent this scenario one must be sure of the software running inside the VoterCard. Fortunately, smart cards support signed applications. Therefore, and because it is possible to know which software is inside the VoterCard by verifying its signature, it is possible to use open source certified software. Of course, we can also have certified applications running on a PC. However, since it is possible for an attacker to take control of the voter's PC, a signed application does not guarantee correct behaviour. On the other hand, it is not possible to take control over the smart card. Therefore, an open source signed application can guarantee correct behaviour.

5 CodeCard

The CodeCard is just a paper card with codes printed on it. There should be one CodeCard per VoterCard so that every voter votes with different codes.

The voter should be the only one with access to the codes of her CodeCard to prevent manipulation of her vote. Consequently we have a problem to solve: how to create the CodeCard, associate it with the VoterCard and give it to the voter without leaking the codes.

We propose to generate each voter's CodeCard within the VoterCard. This is a good option because the CodeCard becomes automatically associated with the correspondent VoterCard and no other entity besides the VoterCard has access to the codes on the CodeCard. However, we still have the problem of how to secretly print the CodeCard, i.e. how to give it to the voter without leaking the codes. We think the best idea is to have a certified CodeCard printing machine, the CodeCard generator interface (CCGI), available at the local authorities' offices. Since the codes are generated inside the VoterCard the CCGI would be very simple. It would consist only of a smart card reader, a keypad (for inserting the PIN of the VoterCard and unlock it) and a small printer. We believe that such a simple hardware could be easily certified and sealed to ensure the secrecy of the codes printed. With the CCGI certified and in place a voter could go to any local authority office to generate a CodeCard for her VoterCard. For privacy reasons the CCGI should be inside a private booth, similar to the ones used for traditional paper based voting.

6 Evaluation

We defend that CodeVoting protects against vote manipulations at the voter's PC under the following assumptions: i) the CodeCard is generated in a secure and controlled environment by the VoterCard (the voter is the only person there), ii) the voter keeps her CodeCard secret, and iii) the correspondence between the candidate and its code (letter) cannot be changed. The last assumption can be achieved by publicly exposing the ballot or by any other technique that prevents a ballot change, such as using an image hard to forge/modify as a ballot.

Under these assumptions changing a vote to a predetermined candidate is virtually impossible because the corresponding vote code is not known by the attacker, and the probability of guessing the correct vote code, with 5 alphanumeric symbols, is 1 in 36^5 , i.e. less than 1 in 60 million.

A random candidate change attack has almost the same probability of success as the previous attack. If we have n candidates running for the election the probability of guessing a random valid vote code is $n - 1$ in 36^5 . However, to prevent an easy denial of service attack the CodeCard should not automatically block when the voter inserts invalid vote codes. Therefore, this limitation allows an attacker to perform a brute force attack to get a random valid vote code. To minimize such an attack simple measures can be used, such as delaying the vote code verification function and/or increasing the length of the vote codes. For instance, with a delay of three seconds the probability of a successful one hour brute force attack is just 1 in 50000, for a vote code with 5 alphanumeric symbols and $n = 11$. Moreover, adding to the three seconds delay an increase of the vote code length to 6 symbols results in a reduction of the attack's success probability to less than 1 in 1 million.

The other possible attack is to fool the voter into believing that she cast a vote, while in reality no vote was cast. To prevent such attack we propose the use of another code to confirm vote delivery. The VoterCard only releases this code after getting a confirmation that the vote was successfully delivered, e.g. could be a message signed by the election server. Therefore, if we use a confirmed vote delivery code with the same length of vote codes, this attack has the same probability of success than the attack of changing the vote to a predetermined candidate. The receipt received by the VoterCard can be stored inside of it to provide a poof of vote delivery, therefore allowing the voter to protest if her vote is not considered for the final tally.

If the voting system does not allow a voter to cast several votes there is also a possible attack to the voters' trust on the voting system. The attack goes as follows: the attacker lets the voter successfully cast a vote and then change the valid confirmed vote delivery code to a fake one. The voter would think that something wrong had happened, however there was nothing wrong. Then the voter would try to vote again and the voting system replies that the voter had already voted and, of course, the voter will protest. If the voting system allows the voter to cast several votes this attack would not be a problem.

Another valuable aspect to evaluate is the implication of CodeVoting in the vote buying/coercion problems. CodeVoting allows the voter to produce a receipt of the vote by giving away the CodeCard to an attacker prior to the election day. On election day, the attacker can demand the voter to vote using a computer controlled by the attacker, e.g. by using a web site controlled by the attacker or a special program develop by the attacker. In this case the attacker will have a vote receipt that proves the voter's vote, therefore enabling vote buying and coercion. We think the best way to prevent such attack is by allowing the voter to vote several times, i.e. update her vote. We believe that the possibility of a vote update in a machine not controlled by the attacker would discourage vote buying and coercion attacks.

7 CodeVoting Issues and Future Work

In this section we will present some issues we have identified with CodeVoting that we will address in future work, namely the capacity of dealing with a large candidate list and the problems raise by the reuse of the CodeCard.

Large candidate lists are a big issue when considering the real application of CodeVoting. As proposed, the CodeCard must have an entry for each possible candidate. If we consider elections with a large number of candidates, lets say above thirty, the size of the CodeCard starts to become to large and usability problems may arise.

Another issue is the CodeCard reuse. If a voter uses the same CodeCard in more than one election it is possible for an attacker to replace the voter's vote by a random one. To be able to perform this attack an attacker must collect the codes used on the previous elections. Additionally, the attacker must also be in control of the PCs used by the voter to vote in each election. If the voter uses

different PCs to vote it will be much harder to perform the attack. We can also make this attack harder by doing all the voting protocol inside the VoterCard, without leaking the voter's identification to the voter's PC. In this case the attacker only has the unlocking PIN to identify the voter.

Of course, the voter can always protect herself against the CodeCard reuse attack by getting a new CodeCard for her VoterCard between elections. However an issue still remains: simultaneous elections. The simultaneous elections issue is a particular case of the CodeCard reuse issue, in which the voter cannot go (or is not convenient to go) to the local authorities and get a new CodeCard. One solution that may work in some particular cases is to use sequential candidate labeling throughout all the simultaneous elections, e.g. instead of using candidate A and B for election 1 and 2, use candidate A and B for election 1 and candidate C and D for election 2. This simple candidate labeling solves the problem of simultaneous elections but can lead to the large candidate list issue.

8 Conclusions

Automatic vote manipulation at client side is one of the biggest dangers that prevent the widespread of Internet voting. We present Code Voting a solution to prevent automatic vote manipulation at client side of a voting application that, at the same time, allows the use of cryptographic voting protocols that protect voters' anonymity and election's integrity at server side.

CodeVoting prevent the manipulation of the voter's vote at three levels: i) prevents the change of the vote to a predetermined candidate, ii) prevents the change of the vote to a random candidate, and iii) prevents vote dropping by a malicious voter's PC.

Besides providing protection against automatic vote manipulation, Code Voting can also provide additional protection to the vote and voter. Namely, the use of CodeVoting in conjunction with a cryptographic voting protocol that runs completely inside the VoterCard may prevent the vote client application, running on the voter's PC, from having access to the clear vote or a message that could directly identify the voter during the voting procedure, therefore providing additional protection to the voter's privacy at the client side of the voting system.

The use of CodeVoting also have implications concerning the vote buying/coercion problem, namely because the CodeCard can be used to get a receipt of the voter's vote. Nevertheless, these implications can be minimized if the voting system allows the voter to update her vote.

Another issues identified in CodeVoting that will deserve our attention in future work are the usability issues concerning elections with a large candidate list, and the issues related with the reuse of the CodeCard, specially in the case of simultaneous elections.

Acknowledgments

We would like to thank the anonymous reviewers and the conference participants for their helpful comments.

References

1. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: CCS 2004: Proceedings of the 11th ACM conference on Computer and communications security, New York, USA, pp. 132–145 (2004)
2. California Internet Voting Task Force: A report on the feasibility of Internet voting (January 2000), <http://www.ss.ca.gov/executive/ivote>
3. Chaum, David: SureVote. September 2007.// International patent WO 01/55940 A1 (02 August 2001), <http://www.surevote.com/home.html>
4. Clarkson, M., Myers, A.: Coercion-Resistant Remote Voting Using Decryption Mixes. In: Workshop on Frontiers in Electronic Elections, Milan, Italy (September 2005)
5. Estonian Internet Voting System (July 2007), <http://www.vvk.ee>
6. Hirt, M., Sako, K.: Efficient Receipt-Free Voting Based on Homomorphic Encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
7. Internet Policy Institute: Report of the National Workshop on Internet Voting: Issues and Research Agenda (March 2001), <http://www.diggo.org/archive/library/dgo2000/dir/PDF/vote.pdf>
8. Jefferson, D., Rubin, A., Simons, B., Wagner, D.: A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE) (January 2004), <http://www.servesecurityreport.org/paper.pdf>
9. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. In: Workshop on Privacy in the Electronic Society, Alexandria, Virginia, pp. 61–70 (November 2005)
10. Kutylowski, M., Zagórski, F.: Coercion-Free Internet Voting with Receipts. In: Workshop on e-Voting and e-Government in the UK. Edinburgh (February 2006)
11. Lee, B., Kim, K.: Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
12. Okamoto, T.: Receipt-Free Electronic Voting Schemes for Large Scale Elections. In: Security Protocols Workshop, Paris, France, pp. 25–35 (April 1997)
13. Rubin, A.: Security Considerations for Remote Electronic Voting Over the Internet. Communications of the ACM 45(12) (2002)
14. Sadeghi, A., Selhorst, M., Stübke, C., Wachsmann, C., Winandy, M.: TCG Inside? - A Note on TPM Specification Compliance. In: STC 2006: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing, Virginia, USA (November 2006)
15. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme A Practical Solution to the Implementation of a Voting Booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
16. UK's National Technical Authority for Information Assurance: e-Voting Security Study (July 2002), http://www.ictparliament.org/CDTunisi/ict_compendium/paesi/uk/uk54.pdf
17. Volkamer, M., Alkassar, A., Sadeghi, A., Schulz, S.: Enabling the Application of the Open Systems like PCs for Online Voting. In: FEE 2006: Proceedings of the Frontiers in Electronic Elections Workshop, Germany (September 2006)
18. Zúquete, A., Costa, C., Romao, M.: An Intrusion-tolerant e-Voting Client System. In: WRAITS 2007: 1st Workshop on Recent Advances on Intrusion-Tolerant Systems, Lisbon, Portugal (March 2007)